福井大学
工学部　研究報告
第33巻　第2号
昭和60年9月

漢字画像データからの漢字ドットパターンの生成

奥村彰二*　佐藤義雄*

Kanji Dot-pattern Generations from the Character Image Data

Shoji OKUMURA and Yoshio SATO

(Received Sep. 2, 1985)

Dot-patterns of Kanji characters are generated by computers from the video image data of printed Kanji characters. Horizontal and vertical line segments of Kanji characters are serially recognized by an edge detection technique of image data processing, and the dot-patterns corresponding to them are plotted for the various sizes of dot-matrices. The dot-patterns of the other segments such as curved lines, slanted lines and dots, are generated by a simple statistical processing of the image data. The algorithm generates the dot-patterns which need only very small interactive corrections.

1　序　　論

近年の日本語ワードプロセッサーやパーソナルコンピューターの発展と普及は，目覚しいものがあり，我々の目に入る文字情報のかなりの部分が，これらからの出力で占められるようになってきている。日本語ワードプロセッサーやパーソナルコンピューターに接続された印字出力機（プリンター）の殆どは，漢字をドットパターンとして印字するものであり，そのドットマトリックスの大きさとしては，24×24のものが現在のところ最も多く採用されている。しかし，より細かく高精度にドットが出力出来るレーザービームプリンターやディジタルな写植機などでは，文字を40×40，48×48，56×56，64×64などのより大きなドットマトリックスで出力されている。このような大きなマトリックスを有効に使用すれば，計算機からの出力として活字印字と見分けがつかないくらい美

＊　情報工学科

しい文字の文章が得られる。一方，計算機と接続できて，普通の日本語タイプライターと同じように活字そのもので印字出来るプリンターが最近比較的安価で製品化されており，今後この方式のものも発展するものと考えられるが，印字速度，字の大きさや字体変更の柔軟性，文字と図形の同時出力の容易さなどの点から，日本語ワードプロセッサーの出力用としては，ドットプリンターが主流をなしていくものと思われる。

さて，24×24の漢字ドットパターン（又は漢字フォント）のセットは，今やリードオンリーメモリーの集積回路（IC）として，安価に入手可能であるが，それより大きな漢字ドットパターンのセットについて，既存のものを自由に複写し，自ら製作した印字システムに組み込むことは出来ない。また今後そうすることが可能になったとしても，印字文字自体を一つの美的な作品と見ることもでき，いろんな特徴ある漢字フォントが存在することは好ましいことであると考えられる。この研究では，活字印字された文字を画像データとしてビデオカメラにより計算機内に取り込み，これを数値的に処理して，任意のサイズの漢字ドットパターンを生成する方法を検討する。一文字毎の漢字画像データを処理して，すべての漢字に対し満足なドットパターが計算機により自動的に生成されることを当面の目標とする。そのためには，結局漢字を構成する字画それぞれを分離し，その形状を認識および分類しなければならない。しかし，図形的に多様な数千の漢字に対しては，簡易的に得られた画像データの不完全性と計算機によるパターン認識の困難性などが相俟って，全く自動的に，完全なドットパターンを生成することは，かなり困難と考えられる。本稿では，計算機がある程度満足のいくドットパターンを生成し，それをグラフィクディスプレイ上に表示し，さらに人がそれに計算機との対話的な修正を行って，満足のいくパターンを得る方法を採用する。計算機との対話的処理により，比較的容易に人の思い通りに修正することは可能であるが，人が計算機に向って仕事しなければならない時間をなるべく短くするには，最初に計算機が修正個所の出来るだけ少ないドットパターンを発生することが必要である。漢字を構成する字画のうち，水平な横線と垂直な縦線を計算機が認識するのは，比較的容易である。この報告では，漢字画像データにおける横線と縦線のドットパターンを認識する方法で，実験したいくつかの漢字について，殆ど対話的な修正なしで，良好な漢字ドットパターンが生成出来ることを示す。

2　ハードウェア構成

この研究で使用されたハードウエア構成のブロック図を図1に示す。

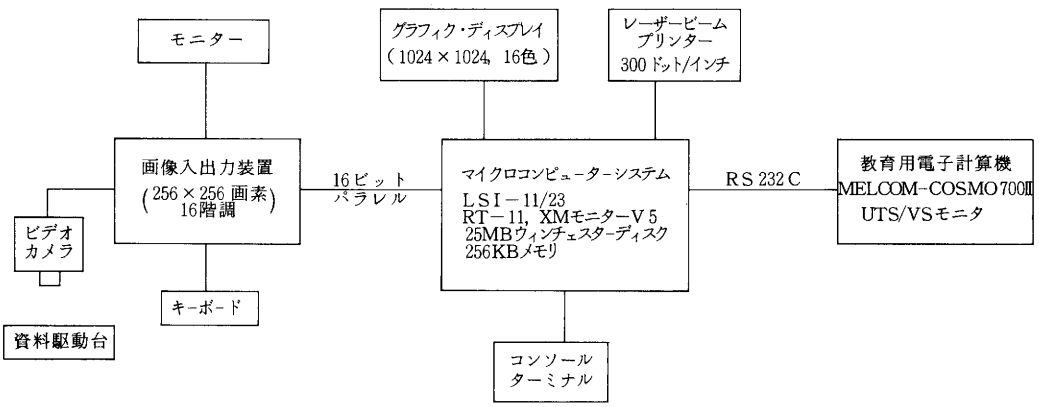

図1　ハードウエア構成のブロック図

画像入出力装置は，昭和57年度の卒業研究[1]により製作されたものを，筆者らがこの研究に合うよう調整改良したもので，ビデオカメラからの1フレームのビデオ信号を，256×256の画素で，16階調のディジタル画像に変換する。またこの装置は，8ビットマイクロプロセッサー8085をもち，簡易モニターが具備されたマイクロコンピュータシステムでもあるので，プログラムをロードしてデータの前処理が可能である。資料駆動台上に乗せられた活字印刷文字から，画像入出力装置のメモリ上に，一文字毎の画像データが取り込まれる。適当にグレイレベルのしきい値を設定して，4ビット／画素のデータを1ビット／画素の2値画像に変換される。この2値画像は，モニター上に常時表示されるので，照明やビデオカメラの設定の不適切などによる不良な画像データの取り込みは避けることが出来る。この画像データは，パラレルインターフェースにより主な計算機処理を行うマイクロコンピュータシステムに転送される。このコンピュータシステムは，1ボード・マイクロコンピュータLSI－11/23（DEC製）に，システムディスクとして25MB容量の$5\frac{1}{4}$インチ・ウィンチェスターディスク（OTARI製）を接続し，RT－11：XMモニター（DEC製）を移植したものである。これに印字出力用として，300ドット/インチのドット密度をもつレーザービームプリンター（LBP－8，キャノン製）と，生成された漢字ドットパターンを，対話的に表示および修正するために，16色，1024×1024画素のカラーグラフィクディスプレイが接続されている。このグラフィクディスプレイのディジタルな回路部は，昭和57年度の卒業研究[2]として，集積回路μPD7220を用いて製作されたものであり，この研究のためこの装置をLSI－11/23のQ－バスに接続したものである。モニターには，インライン型カラーモニタ（C－6919J三菱製）が用いられている。

このコンピュータシステムに，PDP－11用FORTRAN 77を移植し，上記周辺装置に関する手続きは，すべてRT－11のアセンブリ言語であるMACRO－11により，FORTRANサブルーチンとして書かれている。漢字ドットパターンの生成およびそれに対する対話的な修正は，このコンピュータシステムだけで可能であるが，このシステムと情報工学科教育用計算機MELCOM－COSMO 700 Ⅱとの間のデータ転送は常時に出来るので，画像データの蓄積と，より高度な処理のアルゴリズムを並列的に研究するために，画像データをMELCOM－COSMO 700 Ⅱの計算機システムに転送した。

3　文字画像データからのドットパターンの生成

ビデオカメラで写す見本の漢字として，レタリングにより大きく精度よく書いたものを使用出来れば，それに応じて正確な文字画像データが得られると考えられるが，必要な多くの漢字について筆者らがレタリングを行うことは時間的に不可能であり，またそうすることを外部に依頼すれば，非常に高価なものとなってしまう。ここでは，安価に，しかも手軽に文字画像データを得るため，書体辞典およびこの研究用に漢字を写植印字したものを使用した。字体は最も一般的な明朝体を採用した。

前章で述べたようにLSI－11/23のコンピュータシステムに送られる文字画像データは，それ自体256×256のドットパターンである。これをドット密度300ドット/インチ

図2　文字画像データの例

のプリンターで出力すれば，約22mm四方の文字となる。取り込まれた画像データをそのまま出力した例を図2に示す。画像データ特有のノイズが加って，さらに拡大してプリンターに出力してみると，字画の輪郭に凹凸が存在することがわかる。ここでの研究の目的は，これらの256×256のドットパターンから漢字表現として満足な任意サイズのドットパターンを生成することである。生成するパターンのサイズは，計算機でデータが扱い易いように，40×40，48×48，56×56といった8の倍数とする。300ドット/インチのプリンターで，40×40のドットパターンを出力すれば，だいたい10ポイント（約3.5mm四方の文字のサイズ）の標準的な文字サイズに近いものが得られる。議論を進める上で，この40×40のドットパターンの生成を例にとることにする。

3.1　統計的な処理による256×256の画像データからの40×40ドットパターンの生成

まず，単純な統計的処理により256×256の画像データから40×40のドットパターンの生成を行った。このために，256×256の正方形のドット分布を縦横それぞれ40の等間隔のメッシュで分割する。画像データのドットがこのメッシュ線で分割されるときは，適当に比例配分しながら，40×40のメッシュの正方形要素それぞれについて，ドットの数がいくつあるか数え，その数が適当なしきい値より大きいか小さいかによりそのメッシュに相当するドットマトリックスの1要素を1又は0とする。このような変換により生成されたパターンの例を図3に示す。

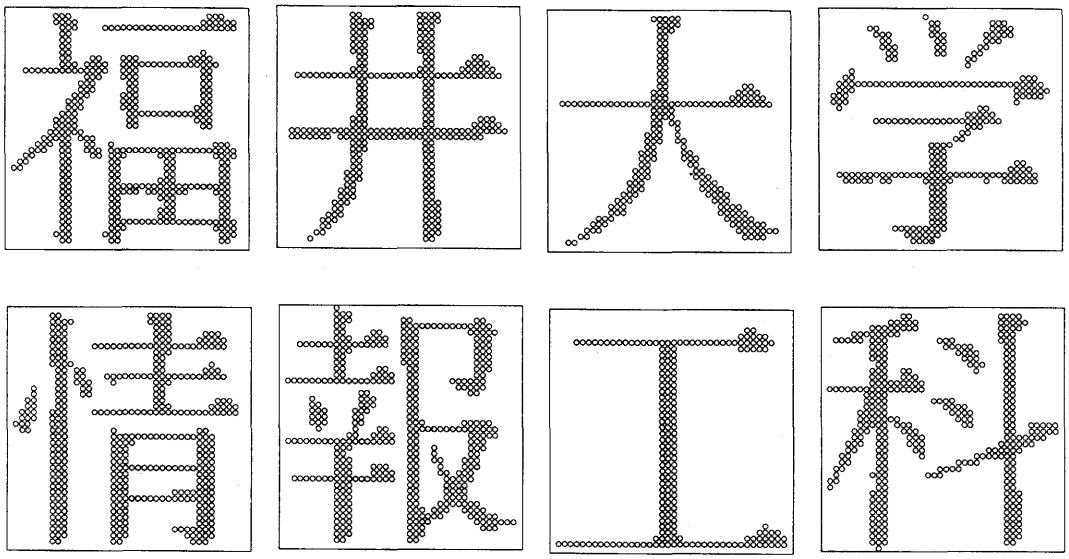

図3　単純な統計的処理のみにより生成された
40×40漢字ドットパターンの例

図3で示されているように，このような方法だけでは満足な漢字ドットパターンは生成されない。特に明朝体の細い横線については，2ドット巾になったり，1ドット巾になったりしている。この例では示されていないが，40×40のドットパターンを生成したとき横線が消えてしまうことも生じた。明朝体の縦線は横線より太く，ドットパターンの変換を行ったとき消えることは起らず，相対的に線巾の変動はより少ないが，2ドットの巾になったり3ドットの巾になったりする。これらの傾向は，画像データを平滑化したり，40×40のメッシュに切ったときその中心部に当る数個の画像データのドットの分布パターンから，漢字パターンを発生しても，同じような結果が得られた。この線巾のばらつきは，画像データでの横線又は縦線が，メッシュ仕切線といろんな相対的位置に存

在するということと，画像データ自体の不正確さとによって生じるものと考えられる。このことを改善するために，水平な横線については，256 × 256 の画像データにおいて，256 本の上下縦方向の画素の並びを走査していき，ドットの連続の巾と位置を検出し，そのドットの並びを，最も距離的に近い40本の水平な仕切り線のうちの一つを選択し，その位置に平行移動した後，同じような統計的処理をする実験も行った。しかし，一本の横線について，ある部分は上の方へ，他のある部分は下の方へ移動する場合が生じ，やはり満足すべき良い結果は得られなかった。

3.2 水平な横線および垂直な縦線の認識によるドットパターンの生成

上記のような単純な統計的処理により，ドットマトリックスを生成するのでは，不十分な結果しか得られないことが明らかとなった。そこで，最も有力な方法として，漢字を構成する字画それぞれを認識し，その形に合ったドットパターンを描いていくことが考えられる。文字を計算機で識別認識することは，パターン認識の研究分野のうち印刷文字又は手書き文字の認識として，多くの研究がなされているが，漢字の字画一つ一つを計算機で分解し認識する研究はあまり報告されていないし，これを厳密に行うとすれば，相当複雑で困難な問題が予想される。明朝体の活字文字についていえば，漢字を構成する字画は，部分的に，水平な横線，垂直な縦線，斜めの線，曲線，点などから成る。このうち，斜めの線，曲線，点などの部分は，3.1 で述べた方法によりドットパターンを発生させると，横線や縦線ほど際立った不満足なところは見い出されない。これらの部分は，その字画を形成する画像データのドットと40×40のメッシュの仕切り線との相対的位置が除々に変化し，一つの字画に関するメッシュ内での画像データのドットの分布は，いろいろな場合が生じるためである。また曲線部は，もともとドットパターンでは表現し難いところであり，ある程度の妥協は本来余儀なくされている。一方，画像データにおける横線および縦線の字画の認識は比較的容易であることが実験より明らかとなった。

（横線の認識） 256 × 256 の画像データは，各々の要素が 0 又は 1 の 256 × 256 のマトリックスであるが，これを 256 個の 256 次の列ベクトルの集りとみなし，各列ベクトルについて，画像処理における一次元的な微分を行う。これは，単に隣り合った要素の差分をその要素とする列ベクトルを発生することである。この処理は，画像のエッジを検出するときに用いられる最も初歩的な手法[3]と考えられる。この 256 個の列ベクトルを加え合わせると，画像の水平方向のエッジの頻度分布を示す一つのベクトルが得られる。このベクトルの要素に適当なしきい値を設定して，横線の有無を判定する。明朝体活字の場合，水平な横線の右端に「とめ」と呼ばれる大きな三角形がついている。そのため，実際には横線の有無を判定するとき，1 から 0 に変化する頻度の大きさを用いて横線の有無を判断する。横線の存在が認められると，次はその横線の始点と終点の位置，また縦線などの他の字画との交わり状況，両端の状態などを検出し，登録する。横線の右端が他の字画と交っているかいないかにより，それに相当するドットパターンを発生するときに，横線の右端に「とめ」の三角形の形でドットを落したり，落さなかったりする。いったん横線の認識が終ると，画像データから横線に相当するドットを消去する。この残りの画像データについて，次の縦線の認識を行う。

（縦線の認識） 縦線の認識の方法は，画像データの行と列を入れ替えて考えれば，既に述べた横線に対する方法と殆ど同じである。ただ，明朝体の縦線の場合，最上部に「おさえ」と呼ばれるこぶがあるので，0 から 1 に変化する縦線の左端に着目する。横線と同じく，縦線の存在を検出した後，その始点と終点の位置，上下端において既に発生した横線のドットとの位置関係などを検出

した後，縦線および必要ならば「おさえ」のドットパターンを発生する。その後画像データから縦線に相当するドットを消去する。

（他の字画のドットパターンの生成） 認識された横および縦の線に相当するドットパターンは元の画像データから除かれているので，残されているドットから3.1で述べた統計的な処理により，追加のドットパターンを生成する。以上3つの方法で生成したドットを合せて，1つの漢字フォントとする。

このような手続きで生成したドットパターンを図4に示す。それらを300ビット/インチのプリンターで出力したものも図4に示してある。

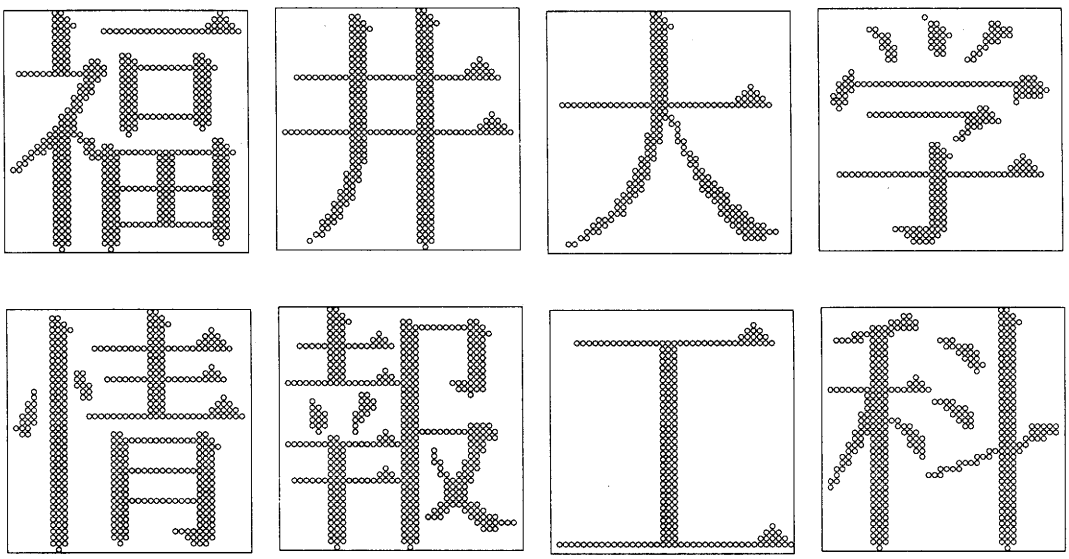

図4　横線と縦線を認識することにより発生したドットパターンとそれを300ドット/インチのプリンターで出力した例

図4に示した実験例では，ほぼ満足のいく漢字ドットパターンが計算機で自動生成されている。生成された漢字ドットパターンはグラフィク・ディスプレイで表示し，さらに不満足な点が見い出されれば，計算機との対話的な応答により修正されることになるが，修正個所は少ないので，他の文字についても短時間で満足すべきドットパターンが得られると考えられる。

4　結語および今後の課題

前章では，40×40のドットパターンの発生を例にとって論じたが，一度横線および縦線の認識がされれば，他のサイズのドットパターンの発生は，極めて容易である。同じような字体にしたければ，そのサイズに比例して発生するドット列の巾や長さを変えればよい。見本が明朝体でもこれらの線を認識した後，ドットを発生するとき横線を太くし，「とめ」の三角形をなくすれば，ゴシック体に似た字が作れそうである。また，少し工夫すれば，教科書体で出力することも可能と考えられる。

ハードウェアの技術が進展し，より安価で高密度，高速のドットプリンターが身近なものになるにつれて，いろいろな字体とサイズの漢字フォントの要求が高まるであろう。多種類の大きな漢字フォントについて，ドットマトリックスのデータそのもので保存するのでは，ファイルが大きくなり，いろいろ困難，不便さが伴なうと予想される。一方，文字を数学的な関数や幾何学的な図形で構成し，計算機によりドットパターンを発生しようとする研究が，既に報告されている[4,5]。特に，英数字については，歴史もあり，最近では，Knuthらの研究[6,7]が有名である。しかし，漢字についての試みは，英数字のようには容易でなく，このようなことに関する研究は少ないようである。もし必要なすべての漢字について，直線やある定義された曲線の集合として，漢字フォントが計算機により美しく表現出来れば，計算機はそれを発生するプログラムとその際必要となる形状を定義するパラメータを含むデータファイルがあればよい。これは，明らかに多くの文字サイズに対してドットパターンでデータを保存するよりは，データが圧縮されることになる。このことを実現するためには，漢字を構成する字画の種類，位置，大きさなどのデータを得ることが必要である。このためには，紙上で描かれた文字について，字画毎にいろいろな幾何学的な測定をしてもよいが，大変時間の要することであるから，計算機で自動的に行わなければならない。ここで行った字画の認識をさらに斜線や曲線部に拡張出来れば，このようなデータの自動的な収集が可能であると考えられる。

この研究を通じて，いろいろ有益な御指適や御検討を下さいました同研究室の長谷川武光助教授に感謝致します。また，漢字フォントの作成に，精力的に御協力いただいている昭和60年度卒業研究生　宇野禎二君に感謝致します。

参　考　文　献

1) 伊藤教之：画像処理入出力装置の試作，昭和57年度福井大学工学部情報工学科卒業論文。

2) 鵜飼信之：高解像度カラーグラフィク装置の設計と製作，昭和57年度福井大学工学部情報工学科卒論文。

3) たとえば，A. Rosenfeld, A.C. Kak: Digital Picture Processing, 2'nd Ed., Vol. 2, Academic Press, 1982, Chapter 10.

4) 山崎，飯島：計算機による標準文字図形の発生，情報処理，Vol.12 (1971), 168 − 175.

5) 山崎，井村：文字輪郭線の円弧と直線とによる近似，情報処理学会論文誌，Vol.26 (1985), 726 − 732.

6) D.E. Knuth: TEX and METAFONT, New Directions in typesetting, American Mathematical Society and Digital Press, Bedford, Mass., 1976, 306.

7) TUGBOAT, The TEX Users Group News Letter, Vol. 1, Providence, R.I., October, 1980.